Prescribed material:

Keet, C.M. (2020). An Introduction to Ontology Engineering.

DeBellis, M. (2021). A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins.

**Outcomes**

After completing this module, you should be able to:

• Demonstrate an understanding of ontologies and their uses

• Critically analyse and evaluate existing ontologies

• Demonstrate an understanding of techniques and methodologies for developing ontologies.

• Use ontology development tools to develop new ontologies and adapt existing ones.

To meet these outcomes, you will cover the following topics:

* Ontologies, terminologies, knowledge representation
* Uses of ontologies
* Semantic Web knowledge
* Semantic interoperability knowledge
* Knowledge of the logical foundations of OWL
* Ontology reasoning
* Bottom-up and top-down development of ontologies
* Using tools for developing ontologies
* Ontology quality and debugging
* Knowledge of domain and foundational ontologies

**The outcomes for the module are covered in six lessons:**

* [Lesson 1: Definition of ontologies, uses of ontologies, and ontology languages](https://mymodules.dtls.unisa.ac.za/mod/lesson/view.php?id=850563).
* [Lesson 2: The Protégé ontology development environment and the Pizza ontology](https://mymodules.dtls.unisa.ac.za/mod/lesson/view.php?id=850761).
* [Lesson 3: Ontology development and foundational ontologies](https://mymodules.dtls.unisa.ac.za/mod/lesson/view.php?id=850810).
* [Lesson 4: Domain ontologies and ontology interoperability examples](https://mymodules.dtls.unisa.ac.za/mod/lesson/view.php?id=855196).
* [Lesson 5: Linking a domain ontology to a foundational ontology](https://mymodules.dtls.unisa.ac.za/mod/lesson/view.php?id=855208).
* [Lesson 6: Representing knowledge expressed as text as an ontology](https://mymodules.dtls.unisa.ac.za/mod/lesson/view.php?id=855282).

**Assessment**:

* COS4840 Ontology Engineering is a continuous assessment module.
* All assessments count towards the final mark for the year.
* There is no formal exam:
  + Two assessments are in the form of practical and theory test
* There is no supplementary exam
* The following formal assessments make up the final mark:
  + 2 theory assignments (compulsory).
  + 1 practical assignment (compulsory).
  + 1 formal test with both practical and theory components.
  + 1 blogging assessment.

**Assessment weights**:

|  |  |  |
| --- | --- | --- |
| Assessment | Type | Weight |
| 1 | Theory assignment | 15% |
| 2 | Theory assignment | 20% |
| 3 | Practical assignment | 25% |
| 4 | Theory test | 15% |
| 5 | Practical test | 15% |
| 6 | Blogging exercises | 10% |

Lesson 2 Overview:

* Demonstrate an understanding of techniques and methodologies for developing ontologies.
* Develop an ontology following the Pizza ontology tutorial.
* Use the **protégé ontology development environment** to build an ontology.

Lesson 3 Overview:

* Critically analyse and evaluate existing ontologies.
  + Evaluate the use of foundational ontologies in ontology development.
  + Distinguish between types of part-whole relationships.

Lesson 3 and Lesson 4 are going to cover assignment 2:

About the Tutorial letter:

Chapter 1:

Conventions:

* **Class, property**, rule and individual names (***Entity***) are written in **Consolas** font.
* **Individual and classes** can also be referred to as **objects**.
* **Highlighted sections** are for tabs, views, menu selections, buttons and text entry.

Chapter 2 – Requirements and the Protégé User Interface:

Chapter 3 – What are OWL Ontologies:

* Ontologies **are used to capture knowledge** about some domain of interest.
* Ontology **describes the concepts** in the domain and also **the relationships that hold** between those concepts.
* We do have different types of ontologies and OWL from **World Wide Web Consortium (W3C)** Is the most recent development in standard ontology language.
* OWL makes it possible **to describe concepts** in an unambiguous manner based on set theory and logic.

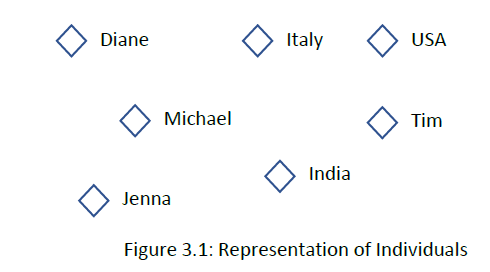
3.1 **Components of OWL Ontologies**:

* An OWL ontology consists of **Classes**, **Properties**, and **Individuals**.
* OWL Ontologies are an **implementation of Description Logic** (DL) which is a decidable subset of First Order Logic.

Note:

* A **class** in OWL is a **set**.
* A **property** in OWL is a **binary relation**.
* An **Individual** is an **element of a set**.

3.1.1 Individuals:

* Individuals represent objects in the domain of interest.
* OWL does not use Unique Name Assumption (UNA) which means that two different names could actually refer to the same individual.
* In OWL, it must be explicitly stated that individuals are the same as each other or of different from each other.
* In this tutorial we represent individuals as diamonds. 
* Individuals are also known as instances. Individuals can be referred to as instances of classes.

3.1.2 Properties:

* Properties are binary relations between individuals. Properties link two individuals together. e.g **hasChild** might link the individual **Michael** to the individual **Oriana**.
* Properties can have inverse. E.g. Inverse of **hasChild** is **hasParent**. A diagram of a diagram

  Description automatically generated

3.1.3 Classes:

* OWL classes are sets that contain individuals.
* Classes may be organized into a superclass-subclass hierarchy which is also known as a taxonomy.
* Classes are represented as ovals, like sets in Venn diagrams.

Chapter 4: Building an OWL Ontology:

* This chapter describes how to create an ontology of Pizzas.
* Set preferences related to the names of new entities.
* In Protégé, any class, Individual, object property, data property, annotation property or rule is referred to as an entity.
* IRI is similar to a URL.

4.1 Named Classes:

* The **main building blocks of an OWL** ontology are classes.
* All empty ontologies contains one class called owl:Thing.
* OWL classes are sets of individuals.

Exercise 4: Create classes: Pizza, PizzaToppin, and PizzaBaze:

4.2. Using a Reasoner:

* Reasoner can be used to check for inconsistencies within your classes.
* One common mistake that new users make is to do a lot of development and then run the reasoner only to find that there are multiple inconsistencies which can make debugging significantly more difficult.
* We will be using Pellet reasoner since we will be writing some rules in SWRL.
* By default, reasoner does not perform all possible inferences because some inferences can take a long time for large and complex ontologies.

4.3 Disjoint Classes:

* Classes are disjoint if no individual can be an instance of more than one of those classes (Pizza, PizzaBase and PizzaTopping).
* In set theory terminology the intersection of these three classes is the empty set (**owl: Nothing**)

4.4 Create Class Hierarchy:

* We use class hierarchy to create multiple classes at once.
* The wizard can add for us the prefix and suffix
* For large ontologies, strict attention to naming of classes and other entities can prevent potential confusion and bugs.

4.6 OWL Properties: